Yocto Project
Autobuilders and the SWAT team

Alexandre Belloni
alexandre.belloni@bootlin.com
Alexandre Belloni

- Co-owner and COO at **Bootlin**
  - Embedded Linux experts
  - Engineering services: Linux BSP development, kernel porting and drivers, Yocto/Buildroot integration, real-time, boot-time, security, multimedia
  - Training services: Embedded Linux, Linux kernel drivers, Yocto, Buildroot, graphics stack, boot-time, real-time

- Open-source contributor
  - Linux kernel **maintainer for the RTC and I3C subsystems**
  - Linux kernel **co-maintainer for Microchip ARM and MIPS processors**
  - **Yocto Project** SWAT team lead

- Living in **Lyon**, France

- alexandre.belloni@bootlin.com

https://bootlin.com/company/staff/alexandre-belloni/
The Yocto Project CI is based on buildbot, a python continuous integration framework.

It is configured using python scripts, available at https://git.yoctoproject.org/yocto-autobuilder2/

This repository lists and defines all the builders (available types of builds) and schedulers.

README.md will help you set up your own autobuilders in a matter of minutes.

More configuration is done through the https://git.yoctoproject.org/yocto-autobuilder-helper repository, especially in config.json.

Nice UI at https://autobuilder.yoctoproject.org/typhoon/

They are necessarily heterogeneous to be able to test on many different distributions: AlmaLinux 8, CentOS 7, CentOS Stream 8, Debian 11, Fedora 35, Fedora 36, openSUSE 15.3, openSUSE 15.4, Tumbleweed, Ubuntu 18.04, Ubuntu 20.04, Ubuntu21.10, Ubuntu 22.04

Also, two different architectures are used: x86 and aarch64
Autobuilders - workers

- x86 workers: most are 28 cores, 56 threads Intel Xeon E5, some are 24 cores, 48 threads with 128 to 384 GB of RAM
- ARM workers: one 64 core Cortex-A72 with 256GB of RAM and one 32 cores APM X-Gene with 128 GB of RAM
- This creates a fair amount of maintenance as the workers need to be configured differently.
- Two workers are specifically reserved for build performance testing (perf-alma8 and perf-debian11)
- About half of the workers now have SSDs
Currently 81 different builders are defined:
- a-full, a-quick, auh, beaglebone, beaglebone-alt, bringup, bringup-fast, build-appliance,
- buildperf-alma8, buildperf-debian11, buildtools, check-layer, check-layer-nightly, docs,
- edgerouter, edgerouter-alt, genericx86, genericx86-64, genericx86-64-alt, genericx86-alt,
- meta-agl-core, meta-arm, meta-aws, meta-intel, meta-mingw, meta-oe, meta-virt, metrics,
- multilib, musl-qemux86, musl-qemux86-64, no-x11, non-gpl3, oe-selftest, oe-selftest-arm,
- oe-selftest-armhost, oe-selftest-centos, oe-selftest-debian, oe-selftest-fedora,
- oe-selftest-ubuntu, pkgman-deb-non-deb, pkgman-non-rpm, pkgman-rpm-non-rpm,
- poky-tiny, qa-extras, qa-extras2, qemuarm, qemuarm-alt, qemuarm-armhost,
- qemuarm-oecore, qemuarm64, qemuarm64-alt, qemuarm64-armhost, qemuarm64-ltp,
- qemuarm64-ptest, qemuarm64-ptest-fast, qemumips, qemumips-alt, qemumips64,
- qemuppc, qemuppc-alt, qemuppc64, qemuriscv32, qemuriscv64, qemuriscv64-ptest,
- qemux86, qemux86-64, qemux86-64-alt, qemux86-64-ltp, qemux86-64-ptest,
- qemux86-64-ptest-fast, qemux86-64-x32, qemu86-alt, qemu86-world,
- qemu86-world-alt, reproducible, reproducible-centos, reproducible-debian,
- reproducible-fedora, reproducible-ubuntu, wic
What is getting built?
Most of the builders build `core-image-sato` using the `poky` distro. Look at `config.json`:

```json
"overrides": {
  "qemu-x86-64": {
    "MACHINE": "qemu-x86-64",
    "TEMPLATE": "arch-qemu",
    "step1": {
      "extravars": [
        "IMAGE_FSTYPES:append = 'wic wic.bmap'"
      ]
    }
  }
}
```
config.json

"templates" : {
  "arch-qemu" : {
    "BUILDINFO" : true,  
    "BUILDHISTORY" : true, 
    "extravars" : [
      "IMAGE_INSTALL:append = ' ssh-pregen-hostkeys'"
    ],
    "step1" : {
      "BBTARGETS" : "core-image-sato core-image-sato-sdk core-image-minimal core-image-minimal-dev core-image-sato:do_populate_sdk",
      "SANITYTARGETS" : "core-image-minimal:do_testimage core-image-sato:do_testimage core-image-sato-sdk:do_testimage"
    },
    "step2" : {
      "SDKMACHINE" : "x86_64",
      "SANITYTARGETS" : "core-image-sato:do_testsdk core-image-minimal:do_testsdkext core-image-sato:do_testsdkext"
    },
    "step3" : {
      "shortname" : "Machine oe-selftest",
      "BUILDHISTORY" : false,
      "EXTRACMDS" : ["${SCRIPTSDIR}/checkvnc; DISPLAY=:1 oe-selftest ${HELPERSTMACHTARGS} -j 15"],
      "ADDLAYER" : ["${BUILDDIR}/../meta-selftest"]
    }
  } }
Autobuilders - builders

- Two parent builders: a-full and a-quick. They will request builds from other builders.
- auh: Tries to upgrade all the recipes to their latest upstream version
- Machine specific builders, building for Yocto Project members machines: beaglebone, beaglebone-alt, edgerouter, edgerouter-alt, genericx86, genericx86-64, genericx86-64-alt, genericx86-alt
- Qemu based machines: musl-qemux86, musl-qemux86-64, qemuarm, qemuarm-alt, qemuarm-armhost, qemuarm-oecore, qemuarm64, qemuarm64-alt, qemuarm64-armhost
- Performance builders, recording the build time and other performance related metrics: buildperf-alma8, buildperf-debian11
- Documentation: docs
- Yocto Project members layers: meta-agl-core, meta-arm, meta-aws, meta-intel
- check-layer, check-layer-nightly: checks whether the included layer are Yocto Project compatible
Autobuilders - builders

- **metrics**: checks the packages for existing CVEs
- **selftests**: oe-selftest, oe-selftest-centos, oe-selftest-debian, oe-selftest-fedora, oe-selftest-ubuntu
- **ptests**: qemuarm64-ptest, qemuarm64-ptest-fast, qemux86-64-ptest, qemux86-64-ptest-fast
- **ltp**: qemuarm64-ltp, qemux86-64-ltp
- **reproducible**: Ensures the generated packages are bit for bit reproducible
- **wic**: tests wic by generating multiple disk images
- **non-gpl3**: builds with `INCOMPATIBLE_LICENSE = '*GPLv3'` and `meta-gplv2`
Any builder can be triggered manually through the buildbot interface

- **a-quick** runs at 1am each day Mon-Sat using the *master* branch
- **a-full** runs at 1am Sun each week using the *master* branch
- **check-layer-nightly** runs each day for *master*
- **metrics** runs at 7am each day
- **check-layer-nightly** runs twice a week for *kirkstone*
- **check-layer-nightly** runs twice a week for *dunfell*
- **AUH** twice a month on 1st and 15th
- **build performance tests** run at 3am, 9am, 3pm and 9pm
- **docs** runs on every commit
An *a-full* build takes from 5 to 9 hours and will load most of the workers.

It is then not practical to start a build automatically for every commit on the master or LTS branches (dunfell, kirkstone)

It is even less practical to do so for every patch series sent on the mailing list

So, build testing is a manual process
Build testing workflow

▶ The process starts by reviewing and collecting patches from the mailing-lists in the appropriate repositories: bitbake, meta-yocto, openembedded-core and yocto-docs

▶ A poky branch is then created from these repositories using combo-layer

▶ This branch is pushed upstream and the autobuilders can be instructed to build a-full using it.

▶ In case of build failures, the incriminated patches are removed from the branch and the process starts over

▶ If the a-full build is successful, a final review is done and patches are merged by Richard Purdie.
The selftest builders run:

- **bitbake-selftest**: tests BitBake and its APIs, including the parser and fetchers.
- **oe-selftest**, skipping the reproducible test: openembedded-core unit tests from `meta/lib/oeqa/selftest/`. Tests devtool, recipetool, archiver, bitbake-layers, CVE checks, INCOMPATIBLE_LICENSE, runqemu, wic
- **oe-pylint**: runs pylint3 when available on the python modules.
Both the regular SDK and the extensible SDK are tested, using the `testsdk` and `testsdk_ext` tasks for the built image.

Tests are in `meta/lib/oeqa/sdk` and `meta/lib/oeqa/sdkext`.

They assume the SDK environment is already setup.

Tests whether the compiler can generate proper binaries for the platform, also tests make, cmake, perl, python,...
The autobuilders don’t have a board farm however, the qemu images are run.

The `testimage` task of the built image is used

- Uses `runqemu` to boot the kernel and use the generated rootfs
- Tests network connectivity, `apt`, `dnf`, `stap`, `systemd`, `weston`, `xorg`

- `-ltp` builders add LTP in the rootfs and runs the test suite.
- `-ptest` builders add `ptest` packages in the rootfs and will run the various tests
  - `ptest`es are unit tests or test suites that are in the upstream release
  - they are always built but have to be explicitly installed
  - e.g.: `openssl-ptest`, `sed-ptest`, `glibc-tests-ptest`, `lttng-tools-ptest`, `python3-ptest`
  - full list is in `meta/conf/distro/include/ptest-packagelists.inc`
Tests - reproducible

The output of the build is bit for bit reproducible (apart from Go and ruby docs)

- Does a first build with Shared State allowed (but not necessarily present)
- Then a second build with Shared State disabled, ensuring binaries will get built
- Both output are then compared
- All package types are tested (ipk, deb, rpm)
- Failures, including binaries and diffoscope output are available at https://autobuilder.yocto.io/pub/repro-fail/
- Results are at https://www.yoctoproject.org/reproducible-build-results/
Some output of the build is saved and archived, in particular:

- **stdio output**, including separate files for warnings and errors.
- **Shared State** as all the autobuilders are populating the same sstate-cache
- **Hash equivalency** is exported through [http://typhoon.yocto.io:8687](http://typhoon.yocto.io:8687)
- **buildhistory** is pushed to [https://git.yoctoproject.org/poky-buildhistory/](https://git.yoctoproject.org/poky-buildhistory/)
- **testresults** are committed to [https://git.yoctoproject.org/yocto-testresults/](https://git.yoctoproject.org/yocto-testresults/)
- **buildstats** from the performance test builds are committed to [https://git.yoctoproject.org/yocto-buildstats/](https://git.yoctoproject.org/yocto-buildstats/)
- Most of those are also available with their logs on [https://autobuilder.yocto.io/pub/non-release/](https://autobuilder.yocto.io/pub/non-release/)
The SWAT team looks at the build failures so none are missed
Its goal is not directly to solve them but to raise attention to the various failures
A specific tool allows to triage those failure: swatbot
https://swatbot.yoctoproject.org/mainindex/swat/
The autobuilders are feeding results to swatbot
Person on SWAT duty will triage failures:
  • For a patch that is under testing and not in master yet, reply on the mailing list, pointing to the build failure.
  • For a commit that is on master, open a bug on bugzilla
  • For an intermittent issue (AB-INT), a bug needs to be open or comment added to an existing bug to allow tracking the frequency of the issue
  • Sometimes the issue has already been handled by the time the failure is triaged
Statistics are also maintained
AB-INT issues - solved

230 tracked AB-INT have been closed

▶ Some are infrastructure related:
  
  • 14551 runtime_test.TestImage.test_testimage_virgl_headless failure

runqemu - ERROR - Failed to run qemu: qemu-system-x86_64: -qmp unix:./.ce0t9x2n,server,wait: info: QEMU waiting for connection on: disconnected:unix:./.ce0t9x2n,server=on
qemu-system-x86_64: egl: no drm render node available
qemu-system-x86_64: egl: render node init failed

This was a permission issue following an OpenSUSE upgrade

  • 14096 perl install race (pod2text)

Couldn't copy cpan/podlators/blib/script/pod2text to /.../usr/bin/pod2text: No such file or directory
Couldn't chmod 755 /.../usr/bin/pod2text: No such file or directory
| /usr/local/oe-sdk-hardcoded-buildpath/sysroots/aarch64-pokysdk-linux/usr/bin/pod2text
| /usr/local/oe-sdk-hardcoded-buildpath/sysroots/aarch64-pokysdk-linux/usr/bin/pod2usage

Reported upstream https://github.com/arsv/perl-cross/issues/75
This ended up being a make issue, happening only on an old version, shipped with ubuntu 16.04 and ubuntu 18.04
AB-INT issues - solved

- 14712 build hangs: host make is not collecting its own children, turning them into zombies
  make issue in version 4.2.1 shipped by centOS, Alma, Stream and openSUSE.
  Reported upstream https://bugs.centos.org/view.php?id=18432 Fixed by disallowing make 4.2.1

- Many were performance related: multiple builds are allowed to run in parallel on a single worker, increasing the load.
  - e.g. Kernel RCU stalls in qemu due to I/Os and host CPU load
  - make load awareness is now used: PARALLEL_MAKE = '-j 16 -l 52'
  - xz limits were lowered:
    
    XZ_MEMLIMIT = '5%
    XZ_THREADS = '8'

  - the qemu rootfs image is copied to tmpfs to avoid I/Os while running
  - the workers are getting switched to SSDs
Some upstream tests are not very well written:

- 14840 AB-INT PTEST ARM: python3 testSockName ptest failure

```
ERROR: testSockName (test.test_socket.GeneralModuleTests)
------------------------------------------------------
Traceback (most recent call last):
  File "/usr/lib/python3.10/test/test_socket.py", line 1380, in testSockName
    sock.bind(("0.0.0.0", port))
OSError: [Errno 98] Address already in use
```

```
test_socket.py

    port = socket_helper.find_unused_port()
    sock.bind("0.0.0.0", port)
```
Some upstream tests are not very well written:

- 14507 AB-INT PTEST: libevent monotonic_prc_fallback_FAILED
  Reported upstream https://github.com/libevent/libevent/issues/1193

regress:
FAIL ../../../libevent-2.1.12-stable/test/regress_util.c:1478: assert(diff.tv_sec == 0): 1 vs 0 util/monotonic_
[monotonic_prc_fallback FAILED]
1/312 TESTS FAILED. (33 skipped)

Fixed by allowing the test to rerun and marking a test failed only when all runs
49 AB-INT are still present. The most pressing or worrying are:

- 14018 efibootpartition.GenericEFITest.test_boot_efi selftest failed
- 14201 Bitbake server intermittent timeout
- 14263 lttng-tools ptest intermittent failure
- 14401 Test unable to login to serial console
- 14522 qemuppc doesn’t shutdown within timeout (serial console issues)
- 14665 prservice.BitbakePrTests.test_import_export_replace_db failure
- 14775 SDK preparation failure: SState: cannot test file://[...]
  TimeoutError(’timed out’)
- 14787 systemdSystemServiceTests.test_systemd_status failure
AB-INT issues - help needed

What needs to be worked on?

▶ Better logging is needed, especially collecting the relevant output when something fails:
  • dmesg output of the target kernel
  • bitbake-cooker.log

▶ More tests are also needed, both for oe-selftest but also upstream tests to run as ptests

▶ Some tests would benefit from being more robust and especially less timing dependent.
Questions? Suggestions? Comments?

Alexandre Belloni
alexandre.belloni@bootlin.com

Slides under CC-BY-SA 3.0